UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Boštjan Primožič

Prototip gradnikov za spletno stran FRI

DIPLOMSKO DELO NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: doc. dr. Janez Demšar

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00482/2009
Datum: 15.10.2009

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **BOŠTJAN PRIMOŽIČ**

Naslov: **PROTOTIP GRADNIKOV ZA SPLETNO STRAN FRI**

PROTOTYPE GADGETS FOR THE WEB SITE OF FACULTY OF COMPUTER AND INFORMATION SCIENCE

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Moderne spletne tehnologije omogočajo razvoj spletnih strani, ki jih lahko uporabnik prikroji svojim željam in interesom. Najbolj znan primer takšne strani je iGoogle, ki uporabnikom omogoča, da na stran razmestijo gradnike (angl. gadgets) z različnimi vsebinami. Odprtokodna tehnologija, na kateri temelji stran, pa uporabnikom z znanjem programiranja omogoča, da razvijajo svoje gradnike. Obenem je mogoče tudi postaviti svojo spletno mesto, ki omogoča vključevanje Googleovih gradnikov oz. gradnikov, ki so jih razvili drugi.

V okviru diplomske naloge z uporabo Googleove tehnologije pripravite prototipe gradnikov, ki bi jih lahko namestili na spletno stran FRI oz. na stran iGoogle in ki prikazujejo informacije, povezane s FRI, denimo novice s spleta, obvestila iz Učilnice, osebni urnik študenta in podobno.

Mentor:

doc. dr. Janez Demšar

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:

prof. dr. Franc Solina

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani/-a **Boštjan Primožič,**

z vpisno številko **63040137,**

sem avtor diplomskega dela z naslovom:

Prototip gradnikov za spletno stran FRI

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom doc. dr. Janez Demšar
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 15.4.2010 Podpis avtorja

Zahvala

Zahvaljujem se svojemu mentorju doc. dr. Janezu Demšarju za zanimivo temo diplomske naloge ter za pomoč in nasvete pri izdelavi le-te.
Prav tako se zahvaljujem tudi vsem, ki so mi stali ob strani in me podpirali v času študija.

Kazalo

Kratice

HTML	HyperText Markup Languange
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
URI	Uniform Resource Identifier
SSH	Secure Shell
SCP	Secure Copy
FTP	File Transfer Protocol
IDE	Integrated development environment
API	Aplication programming interface
XML	Extensible Markup Language
SQL	Structured Query Language
GGE	Google Gadget Editor

Povzetek

Diplomsko delo obravnava razvoj tako imenovanih Googlovih spletnih gradnikov, miniaturnih spletnih aplikacij, ki predstavljajo nov način podajanja informacij uporabniku. Vključimo jih lahko na različna mesta. Najpogosteje je to spletna stran iGoogle ali namizje osebnega računalnika, vendar lahko domujejo tudi na katerokoli drugi spletni strani. Poleg samih gradnikov, ki služijo kot način predstavitve spletne vsebine, so enako pomembne tehnologije v ozadju, ki to vsebino generirajo.
Končni namen je bil izdelava gradnikov, ki bi študentom Fakultete za računalništvo in informatiko nudili enostaven dostop do pogosto rabljenih informacij povezanih z študijem. Trenutno so namreč le-te razdrobljene na mnogih spletnih straneh: od spletne strani fakultete, sistema e-Študent pa do spletne učilnice.
Kot rezultat dela je nastalo devet gradnikov, od katerih vsak omogoča dostop do določenih informacij: tako je zdaj na voljo gradnik, ki vsebuje dnevni ali tedenski urnik, drugi ponuja seznam predmetov za posameznega študenta, tretji obvestila in novice... Upamo, da jih bodo študentje spoznali za koristne in bodo tudi podali predloge za njihovo nadaljnjo izboljšavo in razširitev funkcionalnosti.

Ključne besede:
Googlovi spletni gradniki, Web 2.0, spletne tehnologije.

Abstract

Diploma paper discusses a development of the so-called Google gadgets, miniature web applications that represent a new way of informing a user. They can be incorporated into different places. Most commonly that is iGoogle webpage or a desktop, but they can be placed on any other webpage as well. Apart from the gadgets themselves that serve as a method of presenting web contents, technology in the background which generates these contents is of equal importance.

The ultimate purpose of my diploma paper was to make gadgets that would offer a simple access to frequently used information in connection with studies to students of the Faculty of computer and information science. Currently, these gadgets are scattered on many webpages; from the webpage of the faculty and e-Študent system to the web classroom.

As a result, 9 gadgets have been made, each of them enabling access to particular information: as for example a gadget that involves a daily or a weekly schedule or the one that offers a list of subjects for each individual student, as well as news and notifications. We hope that students will find them useful and will make suggestions for their further improvement and expansion of their functionality.

Key words:
google gadgets, web 2.0, web technology

1 Uvod

Ljudje danes zaradi množice virov informacij in količine le-teh, težimo k združevanju in povzemanju bistva informacij, saj nam edino to omogoča dovolj hiter pregled in izluščenje bistva.

Tako se je tudi v zvezi z informacijami, ki zadevajo študente Fakultete za računalništvo in informatiko pojavila ideja, oziroma želja po drugačnem, bolj dostopnem podajanju le teh. Sedaj so namreč informacije razpršene preko mnogih spletnih strani, kar pomeni slabšo dostopnost in več izgubljenega časa pri iskanju konkretnega podatka.

Zahteva pri izdelavi diplomske naloge je bila združitev razpršenih virov in predstavitev informacij na učinkovit in prilagodljiv način. Kot sredstvo za dosego tega so bili uporabljeni Googlovi spletni gradniki, ki jih lahko poganjamo na različnih mestih. To daje uporabniku pomembno prednost pred klasično spletno stranjo, saj jih lahko poganja na zanj najprimernejšem mestu: jih vključi v spletno stran iGoogle na kateri že ima druge zanj pomembne informacije, kako drugo spletno stran, ali pa jih celo prenese na namizje svojega osebnega računalnika.

Kot rezultat dela je nastalo devet gradnikov, od katerih vsak omogoča dostop do določenih za študenta pomembnih informacij.

Za dosego cilja smo uporabili različne tehnologije, med katerimi so bile ključne:
- Googlovi spletni gradniki
- programski jezik Python
- razširitev spletnega strežnika Apache mod_pyton
- predložni stroj Jinja2

V nadaljevanju diplome bomo tako v drugem poglavju podrobneje predstavimo same gradnike in tehnologije, povezane z njimi. Pri tem najprej opredelimo kaj so gradniki, njihove tipe in zgradbo, nakar sledi opis programskega vmesnika (gadgets API). V drugem delu poglavja se nato posvetimo razvoju in testiranju gradnikov ter sami objavi le-teh. Za konec povemo še, kje uporabniki gradnike najdejo in kje jih lahko uporabljajo.
V tretjem poglavju nato opisujemo postavitev testnega okolja ter v procesu razvoja gradnikov uporabljena orodja in tehnologije.
V četrtem poglavju pa se najprej posvetimo opisu našega dela, pri čemer predstavimo z naše strani razvite strežniške skripte, katere generirajo vsebino gradnikov. Predstavimo tudi vire in

način pridobivanja podatkov za naše gradnike. Na koncu pa predstavimo še razvite gradnike, ter kako le ti delujejo.

2 Googlovi spletni gradniki

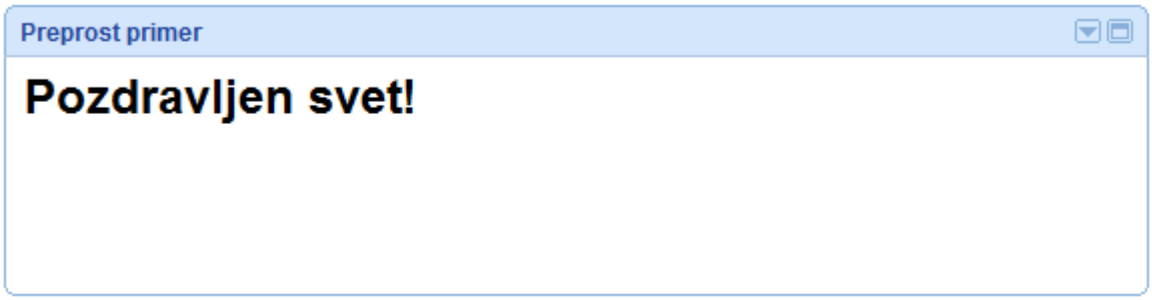
2.1 Kaj so Googlovi spletni gradniki

Googlovi spletni gradniki (ang. Google gadgets), v nadaljevanju gradniki, so mini spletne aplikacije, ki uporabniku nudijo različno vsebino. Tako lahko gradniki vsebujejo igre, seznam opravil, vire novic, koledar,…[1]

Z razvijalskega vidika so gradniki datoteke XML s specifikacijami, ki določajo način procesiranja in prikaza gradnikov in imajo predpisane XML značke. Poleg tega je lahko v datoteki XML, ob sami specifikaciji gradnika, tudi celotna vsebina in programska koda, ali pa zgolj povezava na mesto, kjer se vsebina nahaja.
Vsebina gradnikov je napisana v označevalnem jeziku HTML, dinamičnost pa dodamo z uporabo tehnologije JavaScript ali vključevanjem Flasha [2].

Primer specifikacije preprostega gradnika, katerega vsebino shranimo kot datoteko tipa XML. Rezultat te kode je gradnik na sliki 1.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Module>
  <ModulePrefs title="Preprost primer" height="100" />
  <Content type="html">
    <![CDATA[
      <h2>Pozdravljen svet!</h2>
    ]]>
  </Content>
</Module>
```



1. Primer preprostega gradnika

.

2.2 Tipi gradnikov

Ena od prvih odločitev pri razvoju gradnikov je izbira tipa vsebine gradnika.

Vsebina je lahko tipa HTML ali URL, kar določimo z atributom značke <content> in sicer pri HTML <content type=html> pri URL pa <content type=url href=*URL naslov oddaljene vsebine*>.

2.2.1 Tip HTML

Pri gradnikih tipa HTML vso vsebino pišemo v specifikacije gradnika. Le ta vsebuje HTML, JavaScript, Flash, ActiveX,… omogoča pa tudi vključitev zunanjih JavaScript knjižic.
Ta vrsta je najbolj vsestranska in prilagodljiva in če ste v dvomih kateri tip uporabiti, je ta tip najboljša izbira.

Pri tem tipu gradnikov moramo upoštevati dve pravili.
Prvo zahteva, da mora gradnik vsebovati CDATA sekcijo in da mora vsa koda biti znotraj nje.
Primer:

```
<Content type="html">
  <![CDATA[ HTML here... ]]>
```

CDATA sekcija pove XML razčlenjevalniku, da naj vsebine znotraj nje ne razčlenjuje.

Drugo pravilo pa pravi, da v gradniku ne smemo uporabljati HTML značk <html>, <head>, <body> ampak samo vključimo vsebino, ki bi jo drugače vključili v značko <body>.

2.2.2 Tip URL

Pri gradniku tipa URL se vsa vsebina le-tega nahaja na oddaljeni spletni strani, na katero kaže URL naslov v specifikaciji gradnika. Na oddaljeni spletni strani se tako nahaja vsa HTML, JavaScript in druga koda. Pri tem tipu gradnika ne sme biti v sami specifikaciji le-tega nobene kode, v kolikor pa je, se je ne upošteva.

Pri tem tipu trenutno ni polne podpore za Gadgets API.

2.3 Zgradba gradnika

Specifikacija gradnika je zapisana kot XML dokument z predpisanimi značkami. Prva vrstica v datoteki je standardna in pove, da gre za XML dokument. Sledi značka <Module>, ki pove, da gre za Googlov spletni gradnik, le ta pa sestoji iz treh glavnih delov: nastavitev gradnika, uporabniških nastavitev ter same vsebine.

2.3.1 Nastavitve gradnika

Značka <ModulePrefs> določa sekcijo dokumenta XML, ki vsebuje metapodatke (naslov, podatki o avtorju,…), zahteve dodatnih JavaScript knjižic ter pravila za prikaz gradnika (višina gradnika,…).

Naslednji atributi elementa <ModulePrefs> so podprti v vseh vsebovalnikih:

- title: naslov gradnika, ki se prikaže v naslovni vrstici;
- title_url: URL naslov na katerega lahko kaže naslov;
- description: vsebuje opis gradnika, le-ta pa je v Google direktoriju prikazan ob gradniku;
- author: avtor gradnika;
- author_email;
- screenshot: URL naslov zaslonskega posnetka gradnika v uporabi;
- thumbnail: URL naslov slike, ki služi kot ikona v Googlovem direktoriju.

Sekcija ModulePrefs lahko vključuje tudi gnezdene elemente. Najpogosteje uporabljeni so:

- <Require>: vsebuje ime zahtevane JavaScript knjižice, ki zagotovi želeno dodatno funkcionalnost (<Require feature="dynamic-height"/>);
- <Icon>: določa ikono, katera se prikaže poleg imena v levem meniju, pri čemer je vsebina tega elementa lahko URL naslov, ki kaže na ikono ali pa je ikona sama predstavljena kot base64 kodiran niz;
- <Locale>: določa lokalizacije, ki jih gradnik podpira in vsebuje atribute "lang", "country", "messages" ter "language_direction". Atribut "messages" vsebuje URL, ki kaže na XML datoteko z prevodi nizov, ki jih gradnik vsebuje. Vsak <Locale> element mora vsebovati vsaj enega izmed atributov lang in country.

Uporabnik gradnika do teh nastavitev nima dostopa in jih ne more spreminjati.

2.3.2 Uporabniške nastavitve

Nekateri gradniki morajo uporabniku omogočiti, da sam nastavi določene vrednosti. To dosežemo tako, da vključimo sekcijo <UserPref>, v kateri definiramo uporabniške kontrole. Uporabnik tako na primer lahko določi število prikazanih novic, barvo ozadja,...

Element <UserPref> ima naslednje atribute:

- name: simbolično ime elementa, ki mora biti edinstveno;
- display_name: niz, ki ga vidi uporabnik v nastavitvah poleg tega elementa;
- urlparam: niz, ki predstavlja ime parametra za gradnik tipa URL;
- datatype: tip podatka v tem elementu, ki je lahko string (privzeto), bool, enum, hidden ali list;
- required: argument, ki pove ali je element obvezen ali ne;
- default_value: privzeta vrednost elementa.

Do uporabnikovih nastavitev lahko dostopamo tudi programsko, z uporabo funkcij iz JavaScript knjižice.

Zanimivejša podatkovna tipa sta enum in list. Prvi se v uporabniškem vmesniku prikaže kot padajoči meni izbir, katere določimo z elementi <EnumValue>.
Drugi podatkovni tip pa predstavlja seznam vrednosti, ki jih uporabnik vnese tekom izvajanja gradnika. Do tega seznama prav tako lahko dostopamo programsko, kot do vseh ostalih <UserPref> elementov.

Primer uporabniških nastavitev za izbiro letnika:

```
<UserPref name="letnik" display_name="Letnik" default_value="1" urlparam="letnik" datatype="enum">
<EnumValue value="1" display_value="prvi"/>
<EnumValue value="2" display_value="drugi"/>
<EnumValue value="3" display_value="tretji"/>
<EnumValue value="4" display_value="četrti"/>
</UserPref>
```

2.3.3 Vsebina

Značka <Content> določa sekcijo, kjer se nahaja vsa vsebina in programska logika gradnika. Pri tem značka, odvisno od tipa gradnika, lahko vsebuje dejansko vsebino in programsko logiko ali pa zgolj povezavo na mesto kjer se le-ta nahaja. V kolikor je vsebina že v samem gradniku, jo moramo zapreti v značko <![CDATA[…]]>, ki pove razčlenjevalniku, naj teksta znotraj značke ne obravnava kot XML.

Element <Content> ima naslednje atribute:

- type: določa tip gradnika, pri čemer sta možna tipa url in html, slednji je tudi privzet;
- href: URL naslov oddaljene vsebine;
- view: pri OpenSocial gradnikih določa pogled v katerem je gradnik prikazan;
- preferred_height: začetna višina gradnika;
- preferred_width: začetna širina gradnika.

2.4 Gadgets API

Gadgets API vsebuje več JavaScript knjižic, ki gradnikom omogočajo različne funkcionalnosti. Poznamo temeljno JavaScript knjižico ter specifične JavaScript knjižice, ki jih moramo za uporabo, za razliko od temeljne, posebej navesti.

Od tipa gradnika je odvisen način vključevanja temeljne in specifičnih javaScript knjižic. V primeru da gre za HTML tip gradnika, temeljne knjižice za uporabo ni potrebno posebej navajati, specifične pa navedemo v sekciji <modulePrefs>, z značko require. Na primer *<Require feature="*settitle"/>.
Pri gradnikih tipa URL pa se ime in delna pot do knjižic vključijo kot parametri v URL naslov, ki kaže na oddaljeno vsebino. Na strežniku, kjer se ta vsebina nahaja, moramo to pot izvleči iz URL-ja in konstruirati <script> značko, katera v atributu src vsebuje polno pot do ustrezne knjižice, značko pa nato vključimo v našo HTML kodo.
Naslednji URL naslov http://...&libs=2dhyBXfcpQ8/lib/libcore.js prikazuje, kako je kot parameter vključena temeljna JavaScript knjižica. Niz 2dhyBXfcpQ8 je prstni odtis knjižice in poskrbi za nalaganje pravilne verzije. V html kodo pa nato vključimo naslednjo značko
<script src='http://www.google.com/ig/f/2dhyBXfcpQ8/lib/libcore.js'></script>

2.4.1 Temeljna JavaScript knjižica

Temeljna JavaScript knjižica prinaša splošne funkcionalnosti in jo za uporabo ni potrebno posebej zahtevati. Vsebuje naslednje razrede:
- Prefs;
- io;
- ison;
- util.

Razred prefs vsebuje funkcije, ki omogočajo dostop do uporabnikovih nastavitev in lastnosti ter sporočil gradnika. Za uporabo je potrebno konstruirati objekt razreda Prefs, nakar kličemo želene funkcije tega objekta, kot so na primer getArray(key), getBool(key), getCountry(),…

Razred io zagotavlja funkcije za delo z oddaljeno vsebino. Le te so:

- encodeValues(fields): spremeni vhodni objekt v URL enkodiran niz;
- getProxyUrl(url, opt_params): pridobi posredniško verzijo podanega URL naslova;
- makeRequest(url, callback, opt_params): pridobi vsebino z podanega URL naslova in posreduje odgovor v odzivno funkcijo.

Razred json zagotavlja funkciji za pretvorbo objektov v JSON in obratno. To sta:

- parse(text): razčleni JSON niz in vrne JavaScript vrednost;
- stringify(v): pretvori JavaScript vrednost v JSON niz.

Razred util zagotovi splošno namenske temeljne funkcije. Te so:

- escapeString(str): v vhodnem nizu posebne znake zamenja z ustreznimi ubežnimi sekvencami;
- getFeatureParameters(feature);
- hasFeature(feature);
- registerOnLoadHandler(callback): funkcija, ki pokliče odzivno proceduro, ko je gradnik naložen;
- unescapeString(str).

2.4.2 Setprefs

Ta knjižica omogoča dostop do uporabnikovih nastavitev, ki se nahajajo v sekciji <UserPref>. Omogoča tako branje teh podatkov, kot zapisovanje drugih vrednosti.
Na primer pri gradniku z igro, bi lahko ob koncu igre želeli prebrati seznam najvišjih rezultatov in zapisati število lastnih doseženih točk. To dosežemo programsko, z uporabo primernih get() ali set() funkcij, ki berejo in pišejo v zato določeno polje v uporabnikovih nastavitvah. Tipično je podatkovni tip polja za ta na namen tipa »hidden«, ker ga uporabnik ne vidi in ne more spreminjati.
Pri uporabi te knjižice se moramo zavedati, da <UserPref> sekcija ni namenjena hranjenju velike količine podatkov. Trenutna omejitev je 2kB.

2.4.3 Dynamic-height

Knjižica vsebuje funkcije, ki omogoča gradniku spreminjanje višine.
Privzeta višina gradnika je 200 pikslov, ki jo lahko z uporabo atributa height, v sekciji <ModulePrefs>, nastavimo na poljubno vrednost.
Tako določena višina je statična in se ne prilagaja vsebini gradnika. To lahko rešimo z uporabo vertikalnega drsnika (scrolling="true") ali pa z dinamičnim nastavljanjem višine. Slednje dosežemo z klicem funkcije gadgets.window.adjustHeight, ki ga izvedemo ob spremembi, katera zahteva ponastavitev višine. Primer uporabe bi bil gradnik, ki prikazuje dnevne novice in se njegova višina prilagaja v skladu s številom le-teh.

2.4.4 Settitle

Knjižica omogoča programsko nastavljanje naslova gradnika z uporabo funkcije gadgets.window.setTitle().

2.4.5 Tabs

Knjižica omogoči izdelavo gradnikov z zavihki. Same zavihke zgradimo, jih napolnimo z vsebino in z njimi manipuliramo z uporabo različnih JavaScript funkcij. Tako jih lahko preoblikujemo, določamo njihov vrstni red, poravnavo in vidnost.

Zavihki so sestavljeni iz objektov Tabset, Tab in vsebovalnika vsebine.
TabSet je glavni vsebovalnik in je tabela objektov, ki predstavljajo zavihke. HTML implementacija je tipično izvedena z tabelo, do katere preko API-ja dostopamo z uporabo funkcije gadgets.TabSet.getHeaderContainer().
Tab objekt predstavlja en sam zavihek in se nahaja v indeksiranem polju zavihkov. V HTML-ju je implementiran z uporabo elementa <td>, do katerega dostopamo z uporabo funkcije gadgets.Tab.getNameContainer().
Vsebovalnik vsebine pa vsebuje dejansko vsebino posameznega zavihka. HTML implementacija je praviloma izvedena z uporabo elementa , do katerega dostopamo z uporabo funkcije gadgets.Tab.getContentContainer().

Zavihke ustvarimo tako, da najprej kličemo konstruktor gadgets.TabSet(), kateri ustvari glavni vsebovalnik. Zavihke sedaj dodajamo z uporabo funkcije addTab(). Vsak zavihek ima tri glavne komponente: index, ime in edinstven ID, ki ustreza ID-ju značke , v kateri se nahaja vsebina zavihka.
Ob klicu nam funkcija addTab() vrne DOM id elementa div, katerega nato uporabljamo za

delo z vsebino našega zavihka, medtem ko za manipulacijo samega zavihka uporabljamo njegov indeks.
Knjižica prinaša tudi nekaj že narejenih razredov CSS, ki določajo osnovni izgled zavihkov, sami pa jih lahko prilagodimo po svojih željah.

2.4.6 MiniMessage

Knjižica omogoča enostaven prikaz začasnih sporočil znotraj gradnika, katerih prikaz je kontroliran programsko ali s strani uporabnika.

Nekaj najpogostejših primerov za njihovo uporabo:

- promocija: ta sporočila lahko uporabite za prikaz reklamnih sporočil;
- prikaz statusa: v primeru nalaganja podatkov v ozadju, lahko prikazujemo potek le tega:
- prikaz napak: v primeru da pride v izvajanju gradnika do napake, lahko uporabnika o tem obvestimo, prav pa nam pride tudi tekom razvoja za razhroščevanje;
- Drugo: glede na vsebino gradnika lahko prikazujemo različne posebne informacije.

Glavni tipi teh sporočil so:

- sporočila, ki jih uporabnik zapre z klikom na [x];
- sporočila, ki izginejo po določenem času;
- sporočila, ki jih zapremo programsko ob nekem dogodku.

Sporočila ustvarimo tako, da z klicem konstruktorja gadgets.MiniMessage() najprej ustvarimo MiniMessage objekt, nato pa kličemo metodo tega objekta, ki ustvari ustrezen tip sporočila. Tako z klicem funkcije createDismissibleMessage ustvarimo sporočilo, katerega uporabnik zapre z klikom na [x], funkcija createTimerMessage() ustvari sporočilo, ki izgine po določenem času in funkcija createStaticMessage() ustvari sporočilo, katerega moramo zapreti programsko.

Privzeto se sporočila nahajajo povsem na vrhu gradnika, v primeru večjega števila sporočil, pa se le-ta dodajajo od zgoraj navzdol.
To privzeto lokacijo lahko spremenimo tako, da konstruktorju podamo element HTML, katerega želimo uporabiti za prikaz sporočil (zaželeno je da je ta element). Ta sprememba je globalna in vpliva na vsa sporočila. V kolikor želimo spremeniti lokacijo posameznega sporočila, je potrebno html element, v katerem se bo le-to prikazalo, podati kot parameter ob klicu funkcije.

Izgled sporočil določata dva razreda CSS, ki ju lahko spreminjamo tudi sami in s tem vplivamo na izgled vseh sporočil v tem gradniku. V kolikor želimo spreminjati samo določeno sporočilo, lahko to storimo z nastavljanjem oblikovnih lastnosti elementa HTML, ki ga vrne funkcija, ki je to sporočilo ustvarila.

2.4.7 Flash

Ta knjižica nam omogoči vključitev flash vsebine znotraj gradnika, kar dosežemo z klicem funkcije gadgets.flash.embedFlash().
Ob klicu te funkcije je potrebno podati naslednje parametre:

- pot do datoteke swf;
- ID vsebovalnika, v katerem bo flash vsebina prikazana;
- verzijo swf datoteke;
- opcijske parametre kot objekt, ki lahko vsebuje vse veljavne parametre HTML.

Kot rezultat ta funkcija vrne uspeh ali neuspeh.

Z funkcijo gadgets.flash.getMajorVersion() lahko preverimo ali brskalnik podpira flash razširitve in katero verzijo.

2.4.8 Locked-domain

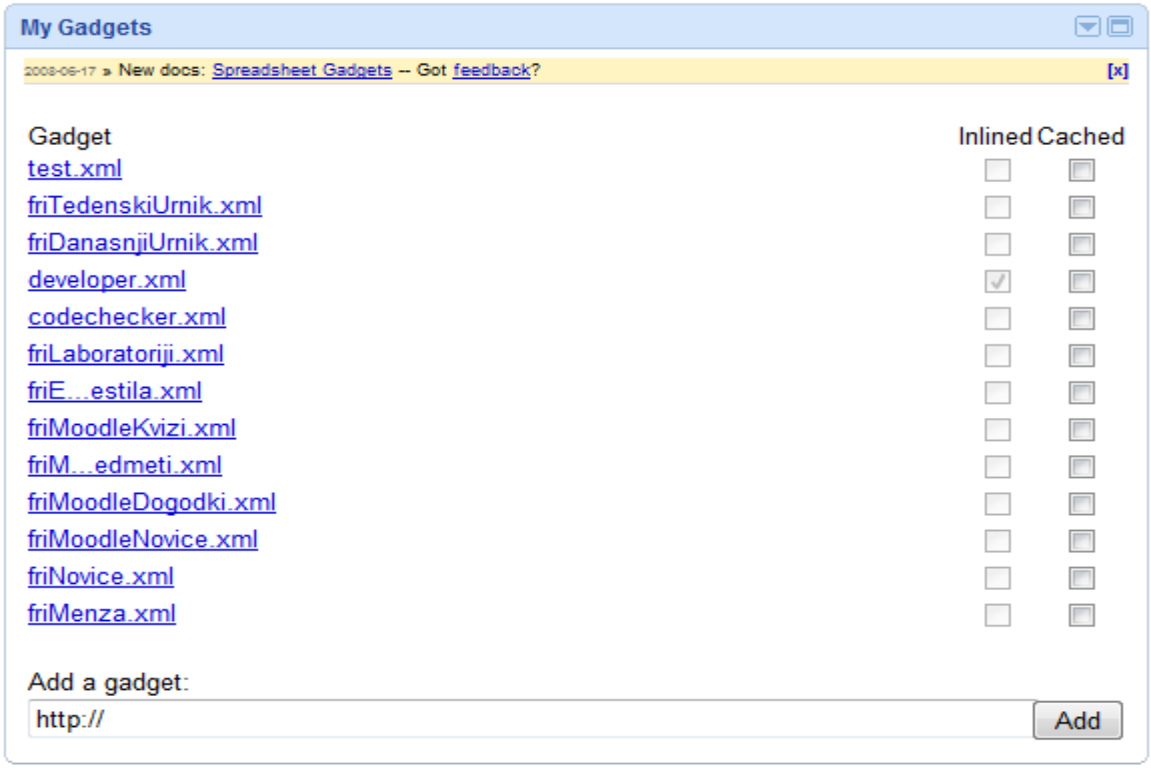
Vključitev te zahteve povzroči izolacijo gradnikov od ostalih, ki tečejo na isti strani. Uporabimo jo lahko le pri gradnikih tipa HTML in je podprta samo na straneh iGoogle ter Google Calendar.
Njena uporaba je priporočljiva, kadar gradnik vsebuje osebne podatke uporabnika.

2.5 Razvoj in testiranje gradnikov

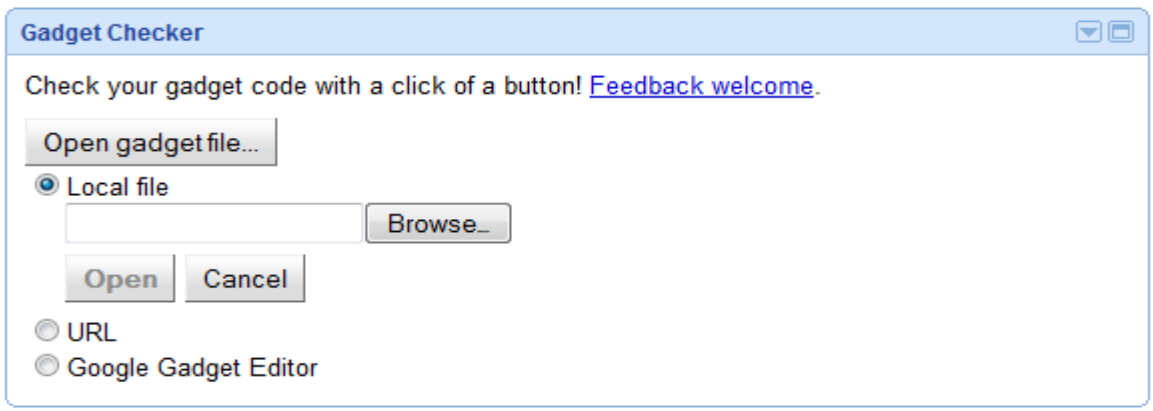
Za pisanje gradnikov lahko uporabimo katerikoli urejevalnik besedila ali pa za prav ta namen, s strani Googla, razvit gradnik Google Gadget Editor (GGE).

Za lažje pisanje gradnikov nam Google nudi tudi različne programske vmesnike API. Ti segajo od osnovnega »gadgets.*« API-ja, v katerega so zajete pogosto uporabljene splošno namenske funkcije, pa tudi do bolj specializiranih, namenjenih lažjemu delu s socialnimi omrežji in drugimi spletnimi stranmi (Google finance, Google maps,…).

Prav tako kot za pisanje, nam Google nudi tudi gradnika za lažje testiranje lastnih napisanih gradnikov. To sta My Gadgets (na sliki 2) in Gadget Checher (na sliki 3).
Prvi nam omogoča enostavno dodajanje gradnikov, ki se nahajajo na spletu, saj je vse kar za to potrebujemo povezava do njihove datoteke XML. Poleg tega prikazuje seznam vseh uporabnikovih gradnikov dodanih na stran iGoogle. Za posamezen gradnik pa omogoča tudi izklop predpomnjenja, ki zagotovi takojšne prilagajanje gradnika na spremembe v njegovi izvorni kodi, kar je še posebej pomembno med razvojem, ko se koda ves čas spreminja.
Drugi, Gadget Checker, pa je namenjen preverjanju pravilnosti gradnika. Pri tem preveri ali je datoteka gradnika dobro formiran XML in če ustreza specifikacijam gradnikov, ki določajo katera polja mora gradnik imeti. Prav tako se preverja pravilnost kode HTML in JavaSripta.



2. Gradnik za pregled in dodajanje naših gradnikov



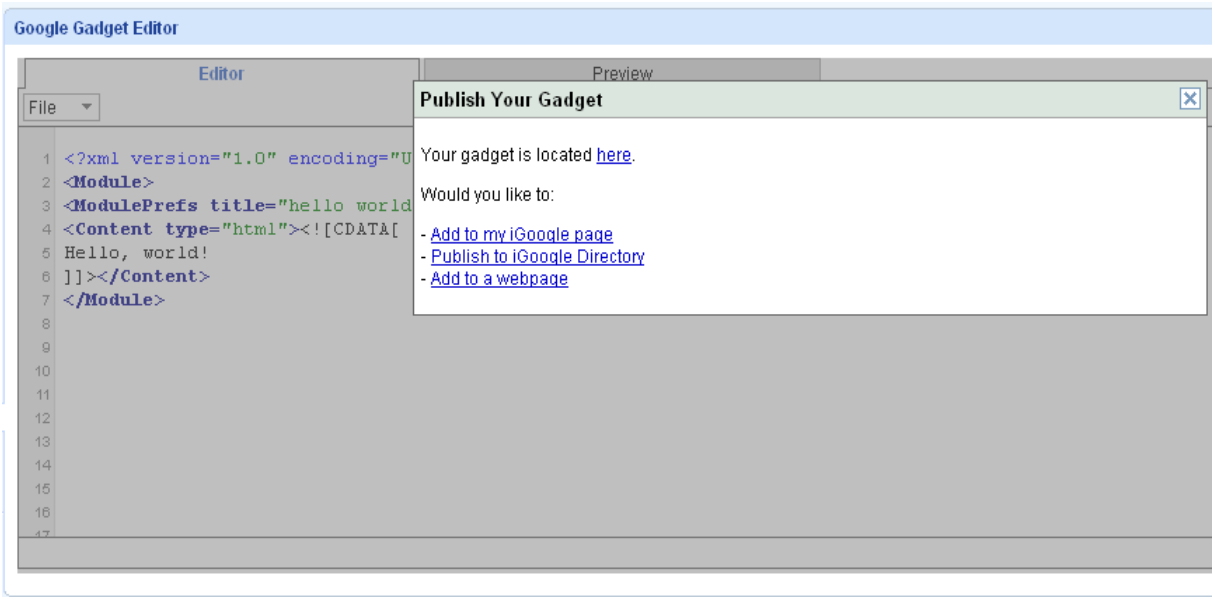
3. Gradnik za preverjanje pravilnosti specifikacije našega gradnika

2.6 Objava gradnikov

Zadnji korak pri razvoju gradnika je njegova objava, ki ga naredi dostopnega ostalim uporabnikom.

Objavimo ga lahko na lastni spletni strani ali pa še bolje v Googlovem direktoriju, kjer bo dostopen najširšemu krogu ljudi. V primeru da smo tekom razvoja uporabljali GGE, kot je prikazano na sliki 4, pa je objava gradnikov še toliko lažja.

Pred objavo je potrebno gradnik dobro pretestirati, pri čemer nam pomaga prej omenjeni gradnik Gadget Checker, in če pričakujemo množično uporabo, tudi optimizirati količino prenesenih podatkov. Prav tako mora njegova specifikacija vsebovati vse zahtevane elemente.



4. Objavljanje gradnika z uporabo GGE

2.7 Kje gradnike lahko uporabimo

Registrirani uporabniki Googlovih storitev lahko objavljene gradnike dodajo na svojo stran iGoogle, gmail, Google Calender in druge. V primeru, da ima uporabnik nameščen Google Desktop, pa jih lahko poganja tudi na namizju svojega računalnika. Gradnike lahko uporabniki dodajo tudi na svojo spletno stran, pri čemer imajo možnost določene prilagoditve izgleda gradnika, s čimer lahko dosežejo boljšo integracijo le-tega s spletno stranjo.

Uporabniki gradnike najpogosteje najdejo v Googlovem direktoriju (na sliki 5), od koder jih lahko namestijo na želeno mesto z nekaj preprostimi kliki. V primeru dodajanja na lastno spletno stran, pa na ustrezno mesto le-te, samo vključijo kodo, ki jo je pripravil Google.

5. Izsek strani Google direktorij

3 Uporabljena orodja in tehnologije

3.1 Postavitev testnega okolja

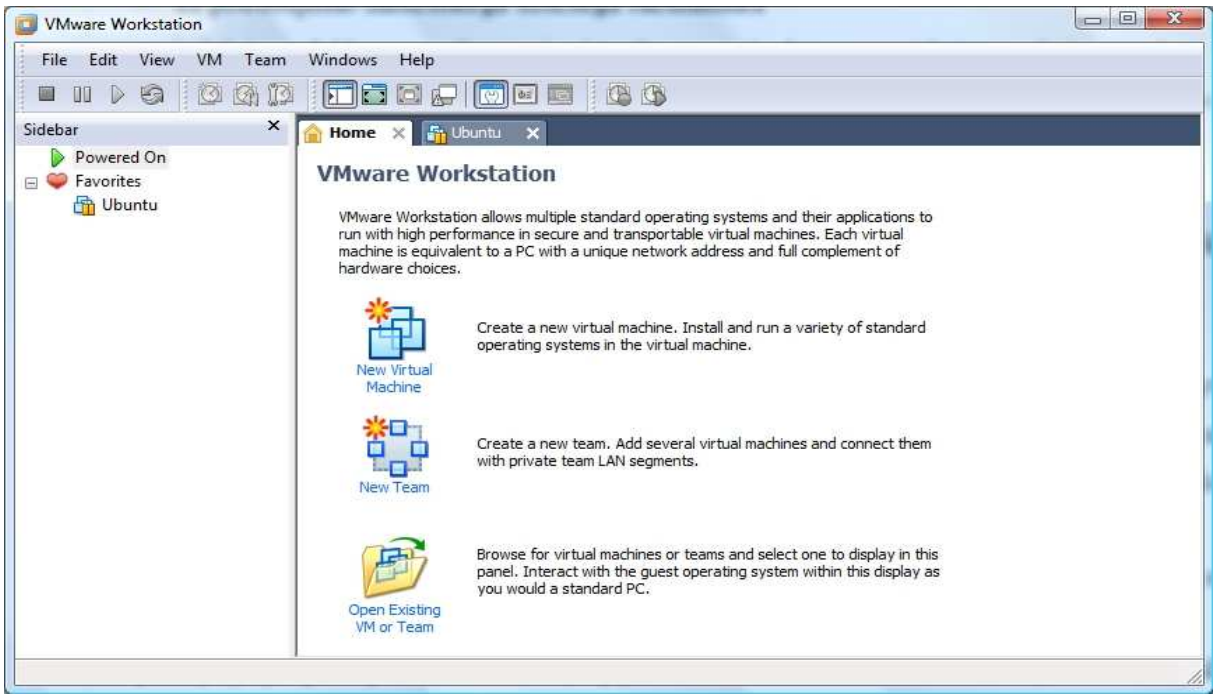
Kmalu po začetku dela na diplomski nalogi se je pokazala potreba po vzpostavitvi okolja, ki nam bo omogočalo sprotno testiranje rezultatov dela in bo na koncu, ko bo delo zaključeno, služilo tudi kot strežnik za naše gradnike.

3.1.1 Virtualizacija testnega okolja

Samo testno okolje smo se odločili postaviti v virtualno okolje. Ta način nam prinaša precej prednosti:

- ne potrebujemo namenskega fizičnega računalnika;
- izdelujemo lahko posnetke stanja virtualnega stroja, kar nam pride še posebej prav ob začetni konfiguraciji, saj lahko v primeru, da kaj ne deluje, sistem enostavno povrnemo v prejšnje stanje;
- virtualni stroj lahko prenesemo na drug računalnik - recimo iz domačega računalnika na strežniško rezino ali obratno. Tako imamo lahko na razvojnem sistemu natančen posnetek produkcijskega sistema.

Za virtualno okolje smo uporabili VMware Workstation 7 [5,6], ki je prikazan na sliki 6. Podjetje VMware, ki ga izdeluje, je prvo in še vedno najuspešnejše podjetje na področju virtualizacije sistemov. Njihova rešitev nam omogoča enostavno izdelavo virtualnega stroja, njegovo upravljanje ter nadvse uporabno izdelavo posnetkov stanja. K naši izbiri je odločilno pripomoglo tudi to, da bo virtualni stroj kasneje tekel na strežniški rezini z nameščenim VMware ESXi istega podjetja. Sam program je sicer komercialen, tako da smo uporabili brezplačno preizkusno različico z omejenim tridesetdnevnim delovanjem. Kljub tej omejitvi, pa je program povsem zadoščal, saj je služil le toliko časa, da smo izdelali virtualni stroj, vanj namestil svoje testno okolje ter ga nastavil v skladu s potrebami za izvedbo diplomskega dela. Kasneje se je kopija virtualnega stroja enostavno prenesla na strežniško rezino, kjer nadalje opravlja svoje delo.



6. Program VMware workstation 7

3.1.2 Operacijski sistem

Za operacijski sistem testnega okolja smo izbrali Ubuntu Linux [8], ker je prosto dostopen. Prosta dostopnost pomeni, da je operacijski sistem brezplačen, poleg tega pa je zelo prilagodljiv, dokaj varen ter ima močno uporabniško skupnost na spletu. Slednje se je med kasnejšim nastavljanjem sistema izkazalo kot velik plus, saj smo rešitve vseh problemov, na katere smo naleteli, zlahka našli na spletu, predvsem na spletnem forumu Ubuntu [7].

Sama namestitev sistema znotraj virtualnega okolja je enostavna, izbrati moramo le vir namestitve; ta je lahko datoteka ISO ali pa optični pogon. Namestitev se potem nadaljuje enako kot na fizični računalnik. Po končani namestitvi sistema sledi še namestitev posebnega gonilnika VMWare tools, ki izboljša delovanje gostujočega sistema in njegovo integracijo z gostiteljskim sistemom (prosto prehajanje miške, kopiranje in lepljenje…).

Operacijski sistem smo zaradi naših potreb in želje po čim manjši velikosti namestitve nekoliko prilagodili. Odstranili smo množico programskih paketov in dodatkov, ki na strežniškem sistemu niso potrebni. Manj nameščenih programov hkrati pomeni boljšo odzivnost in hitrejše delovanje sistema, kar je za strežnik nadvse pomembno.

Sledilo je še nameščanje ostalih programskih paketov, ki so potrebni za celovitost našega testnega okolja. Ti programski paketi so seveda nameščeni znotraj virtualnega okolja, mednje spadajo naslednji:

- spletni strežnik Apache,
- razširitev spletnega strežnika mod_python,
- podatkovna baza MySQL,
- python modul python-mysqldb,
- SSH strežnik openssh-server.

3.1.3 Spletni strežnik

Spletni strežnik je program, ki odjemalcem pošilja spletne strani z uporabo protokola HTTP. Je torej pomemben element pri naši diplomski nalogi, saj streže vsebino, ki jo prikazujemo z uporabo gradnikov.

Za strežnik smo izbrali Apache HTTP Server, saj je le-ta najbolj razširjen spletni strežnik na svetu z več kot 50% tržnim deležem, je odprtokoden in kot tak brezplačen za uporabo [9, 10].

Njegove glavne značilnosti so:

- odprtokodnost in prenosljivost
- zanj obstaja veliko dodatnih modulov, ki razširjajo osnovno funkcionalnost, za namene te naloge najbolj zanimiv je modul mod_python, ki omogoča poganjanje skript, napisanih v programskem jeziku python
- velika razširjenost in posledično močna uporabniška skupnost
- varnost in zanesljivost

Namestitev v okolju Ubuntu Linux je sila preprosta in jo opravimo z ukazom:

```
sudo apt-get install apache2
```

Po namestitvi se strežnik zažene sam, njegovo delovanje pa preverimo tako, da odpremo spletni brskalnik in vtipkamo IP naslov računalnika na katerem se nahaja. V kolikor je bila namestitev uspešna, nas pričaka pozdravno sporočilo strežnika Apache.

7. Sporočilo strežnika Apache, ki naznanja njegovo delovanje

3.1.4 Mod_python

Sam spletni strežnik pa za naše testno okolje ni dovolj. Namreč, ker se bo koda HTML, torej vsebina naših gradnikov tvorila dinamično, potrebujemo podporo izvajanju strežniških skript. Skripte so lahko napisane v različnih jezikih. Priljubljena izbira spletnih razvijalcev so ASP, PHP, python in ruby, med redkeje uporabljanimi pa je tudi perl.
Sami smo se odločili za python, ker gre za moderen in zmogljiv jezik. Za izvajanje v njem napisanih strežniških skript je potrebno strežnik Apache razširiti z ustreznim modulom. Sami smo uporabili modul mod_python, kateri je za naše potrebe povsem zadostoval.

Kako deluje mod_python v Apachu

Spletni strežnik Apache vsak zahtevek obdela v fazah. Pri tipičnem zahtevku statične datoteke se tako zgodijo naslednje faze:

1) zahtevani URI se pretvori v lokacijo in ime datoteke;
2) datoteka se prebere in pošlje odjemalcu;
3) zahtevek se zapiše v dnevniško datoteko.

Vsako fazo izvede upravljalec (handler) - funkcija strežnika Apache. Upravljavci izvajajo samo zelo osnovne operacije ali celo ne delajo nič, dodatno funkcionalnost pa prinesejo Apache moduli - kakršen je tudi mod_python. Mod_python ponuja Apachu najrazličnejše upravljavce, čeprav privzeto ne opravljajo nobene funkcije. Za njihovo delovanje jih moramo najprej omogočiti, tako da jih navedemo v nastavitveni datoteki spletnega strežnika. Vsi upravljavci se pričnejo z besedico „Python“ in končajo s „Handler“ (primer: PythonAuthenHandler) ter povežejo posamezno fazo procesiranja zahtevka z ustrezno funkcijo napisano v pythonu. Glavni namen modula mod_python je, da deluje kot dispečer med Apachovimi upravljavci in funkcijami, ki smo jih sami napisali v programskem jeziku python.

Najpogosteje uporabljen upravljalec je PythonHandler. Ker nima imena, oziroma je le-to prazno, se ga včasih imenuje tudi splošni upravljalec. Privzeta akcija Apacha zanj je branje datoteke in pošiljanje njene vsebine odjemalcu. V našem primeru pa bo Apache naložil modul, ki je naveden poleg ukaza PythonHandler in ga začel izvajati, rezultat pa bo poslal nazaj odjemalcu.
Pisanju lastnih upravljavcev se lahko izognemo, če uporabimo že pripravljen „publisher handler“, s čimer se lahko osredotočimo na hiter razvoj aplikacije.
Publisher handler omogoča dostop do spremenljivk in funkcij znotraj pythonovih modulov, prek naslovov URI. Sam algoritem deluje tako, da:

- na podlagi naslova URI poišče in naloži ustrezen modul, privzeto je to „index“
- preostali del naslova URI do začetka poizvedbenih podatkov (query data) se porabi za lociranje objekta znotraj modula. Privzeto je ta objekt „index“
- poizvedbeni podatki se v primeru, da je objekt funkcija, preslikajo v parametre te funkcije

V primeru funkcije se kot niz vrne rezultat njenega izvajanja, če pa gre za spremenljivko ali razred, potem se vrne njena vrednost oz. predstavitev razreda v obliki niza. V kolikor modula ali objekta ne najde, oziroma se slednji začne z podčrtajem (podčrtaj v pythonu označuje da gre za nekaj, kar naj bi bilo privatno) pa se klientu vrne odgovor, da zahtevanega ni moč najti oziroma, da dostop ni dovoljen [11].

Namestitev

Mod_python namestimo z ukazom

sudo apt-get install libapache2-mod-python

Po končani namestitvi je potrebno urediti še nastavitve strežnika Apache. Slednje storimo tako da v datoteko */etc/apache2/sites-available/default* dodamo sledeči zapis:

```
<Directory /var/www/>
        AddHandler mod_python .py
        PythonHandler mod_python.publisher
        PythonDebug On
</Directory>
```

Pomen direktiv je naslednji:

- prva pove, da mora biti vsak zahtevek, ki se nanaša na datoteko z končnico .py, sprocesiran s strani modula mod_python;
- druga pove, naj mod_python za procesiranje splošnega handlerja uporabi vgrajeni

publisher handler;
- tretja direktiva pa v primeru napake povzroči, da se sporočila o napakah pošljejo odjemalcu in ne le zapišejo v dnevniško datoteko.

Za uveljavitev sprememb je potrebno ponovno zagnati proces strežnika Apache, kar storimo z ukazom:

sudo/etc/init.d/apache2 restart

Preizkus delovanja

Ali po namestitvi vse deluje pravilno lahko preizkusimo z preprosto skripto. V imeniku /var/www/ ustvarimo datoteko test.py in vanjo prepišemo sledečo kodo.

```
from mod_python import apache

def index(req):
   req.content_type = 'text/plain'
   req.write("Pozdravljen svet!")
```

Sedaj v brskalnik vtipkamo naslov http://localhost/test in če je bila namestitev uspešna dobimo na zaslon sporočilo »Pozdravljen svet!«.

3.1.5 Podatkovna baza MySQL in modul python-mysqldb

Za testno okolje so nujno potrebni podatki, nad katerimi delamo. V našem primeru je del teh podatkov anonimizirana podatkovna baza spletne e-učilnice, ki se uporablja na fakulteti. E-učilnica temelji na odprtokodnem sistemu Moodle [24], ki za hrambo podatkov uporablja podatkovno bazo MySQL. Sama podatkovna baza MySQL je prav tako na voljo pod odprtokodno licenco in je zelo priljubljena predvsem pri različnih spletnih aplikacijah [12]. Namestitev same podatkovne baze je zelo enostavna in jo opravimo z ukazom:

sudo apt-get install mysql-server

Pred končanjem namestitve je potrebno še vnesti geslo korenskega uporabnika, potem pa je pripravljena za uporabo.

Da lahko iz programskega jezika python dostopamo do podatkovne baze MySQL potrebujemo modul python-mysqldb. Modul nam prinaša programski vmesnik API za delo z samo podatkovno bazo [13, 14]. Namestimo ga z ukazom:

sudo apt-get install python-mysqldb

Zadnji korak je bil kreiranje baze in uvoz prej izvoženih in anonimiziranih podatkov. To storimo v ukazni konzoli MySQL z sledečimi ukazi:

mysql> CREATE DATABASE moodle;
mysql> ALTER DATABASE moodle DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;
mysql --user=XXXXX --password=XXXXX moodle < /pot/do/datoteke.sql

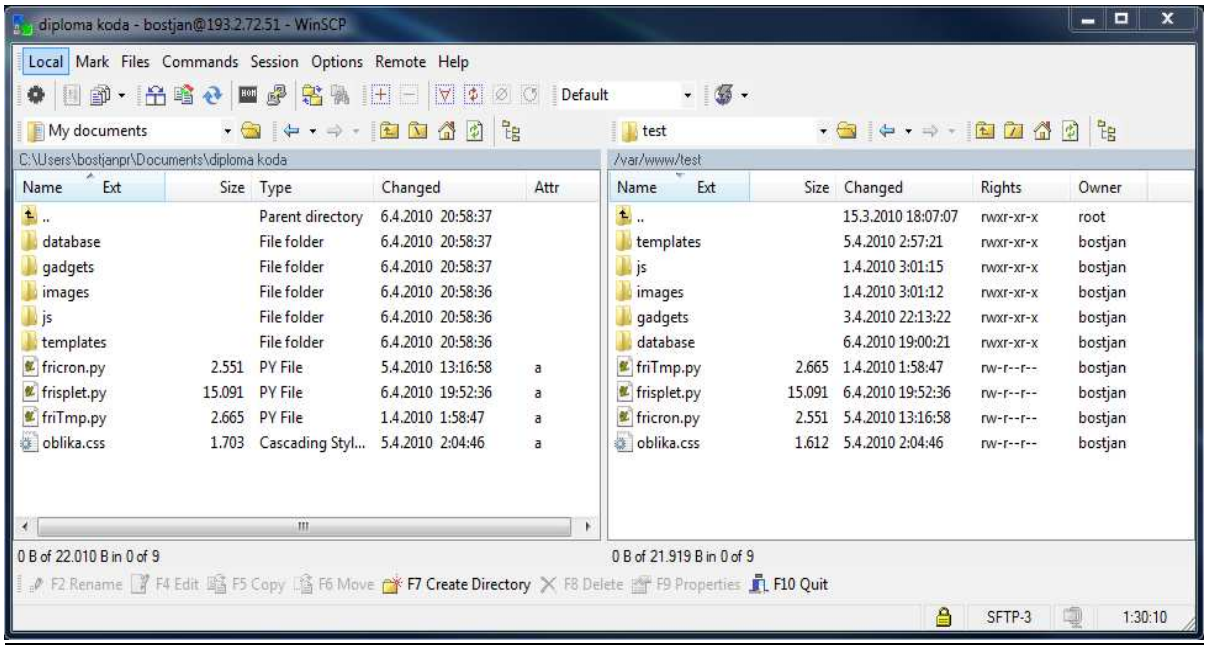
S tem je bila namestitev in priprava podatkovne baze ter z njo povezanega modula končana, sami podatki pa povrnjeni in pripravljeni za uporabo.

3.1.6 Dostop do testnega okolja

Za konec priprave testnega okolja je bilo potrebno urediti še oddaljeni dostop do njega, saj bo le-ta pozneje tekel v virtualnem okolju na strežniški rezini. Za oddaljen dostop se najpogosteje uporablja protokol SSH (secure shell), ki je sodobnejši nadomestek telneta. V primerjavi z njim je predvsem varnejši saj ponuja močno šifriranje povezave [22].
Na strežniku smo tako namestili strežnik Open SSH, kar storimo z ukazom:
sudo apt-get install openssh-server.

Kot odjemalca za dostop do strežnika preko SSH uporabljamo program PuTTY, ki je za te namene najbolj pogosto uporabljan program, predvsem zaradi tega, ker je brezplačen in izjemno prenosljiv [23].
Na strežnik se povežemo tako, da poženemo omenjeni program, vpišemo IP naslov želenega strežnika in izberemo protokol (SSH je privzet) ter kliknemo „Open“. Ko se povezava vzpostavi, se le še prijavimo s svojim uporabniškim imenom ter geslom.

Za nalaganje datotek na strežnik smo uporabljali program WinSCP, ki za povezovanje prav tako uporablja protokol SSH in z uporabo protokola SCP omogoči prenašanje datotek preko SSH povezave. V primerjavi z FTP-jem tako dosežemo varen prenos podatkov, saj SSH poskrbi za avtentikacijo in enkripcijo, poleg tega pa se pri prenosu ohranijo osnovni atributi datotek (pravice, časovni žig) [15, 16]. Prikaz programa med delovanjem je na sliki 8.



8. Program WinSCP

3.2 Razvojna orodja in tehnologije

V tem poglavju so predstavljene vse tehnologije in orodja, ki sem jih uporabil v procesu izdelave gradnikov. Za vsakega so napisane glavne značilnosti, in kje ter kako so bila uporabljena v procesu izdelave.

3.2.1 Programski jezik Python

Python je splošno namenski skriptni programski jezik, ki ga odlikuje velika berljivost programske kode, kar je bil tudi eden izmed ciljev pri snovanju jezika samega. Programiranje v pythonu je hitro, ker je jezik visokonivojski, z močnimi vdelanimi podatkovnimi strukturami. Poleg tega vključuje bogato zbirko že napisanih modulov za skoraj vsa opravila, če pa karkoli pogrešamo, pa iskani modul skoraj zagotovo dobimo na internetu povsem brezplačno. Posledično pisanje programa dostikrat pomeni zgolj zlaganje že pripravljenih delčkov.
Python je objektno usmerjen, saj je v njem vse objekt, od celega števila pa do funkcij, vendar za razliko od Jave ne vsiljuje objektnega programiranja. Podpira namreč še dve paradigmi: funkcijsko in proceduralno programiranje.
Ker gre za dinamičen skriptni jezik, na račun tega trpi hitrost samega izvajanja programa. Sama koda se sicer neopazno prevaja v vmesno kodo, kljub temu pa hitrost izvajanja ne

dosega hitrosti sodobnih javanskih programov. Če je hitrost nujna, lahko kritični del kode implementiramo v C-ju, s katerim se Python lepo povezuje [17, 18].

Glavne značilnosti Pythona:

- enostavnost
- berljivost kode
- prenosljivost
- hiter razvoj
- bogat nabor knjižic za vsa mogoča opravila
- brezplačno dostopen za razvijalce in uporabnike

Python smo uporabljali za razvoj strežniških skript, ki skrbita za pridobivanje podatkov ter generiranje vsebine gradnikov.

3.2.2 Integrirano razvojno okolje Eclipse

Razvoj aplikacij se pri uporabi sodobnih razvojnih okolij močno poenostavi. Skripte v Pythonu sicer lahko pišemo z vsakim, še tako preprostim urejevalnikom teksta, vendar nas to prikrajša za mnogo prednosti, ki jih nudi IDE. Sodobno razvojno okolje praviloma ponuja:

- preverjanje sintakse,
- razhroščevanje kode,
- hitro navigacijo po kodi,
- prikaz metod objektov,
- prikaz pomoči in definicij funkcij ter objektov.

Eclipse [19] je integrirano razvojno okolje IDE, ki je bil prvotno ustvarjen za pisanje aplikacij v programskem jeziku Java. Skozi razvoj je dobil mnogo dodatnih vtičnikov, ki so prinesli podporo mnogim drugim programskim in skriptnim jezikom, kot so C, C++, PHP, Python… Za pisanje programov v Pythonu je potrebno namestiti vtičnik PyDev, ki razvojno okolje prilagodi pisanju programov v tem jeziku, prav to pa smo storili tudi mi.

3.2.3 Toad for MySQL

Toad for MySQL je brezplačen program za delo in upravljanje podatkovne baze MySQL. Omogoča administracijo podatkovne baze, pregledovanje vsebine in pisanje ter testiranje SQL poizvedb [20].
Uporabljali smo ga pri delu z podatkovno bazo sistema Moodle in sicer za raziskovanje sheme le-te in za kasnejše pisanje poizvedb.

3.2.4 Jinja2

Jinja2 [4] je tako imenovani predložni stroj (ang template engine). To je program, ki je zasnovan za procesiranje predlog in pripadajočih podatkov z namenov stvaritve izhodnega dokumenta. Prednosti so:

lažja organizacija kode v različne plasti: ločitev predstavitvene plasti od kontrolne in modela (npr. koncept MVC);

izboljšanje produktivnosti;

lažja delitev dela: nekdo pripravi predlogo, ki določa videz, drugi pa podatke, ki jih bomo prikazovali.

Jinja2 je splošno namenski predložni stroj za programski jezik python, v katerem je tudi napisana. Glavne značilnosti so :

- nastavljiva sintaksa: spremenimo lahko razmejitve, ki označujejo kontrolne stavke v predlogi tako, da se ti bolje skladajo z označevalnim jezikom dokumenta;
- hitrost: le malo počasnejša od direktne manipulacije nizov;
- enostavno razhroščevanje: v Python se integrira tako, da omogoča razhroščevanje kot pri normalni python kodi;
- varnost: omogočimo lahko izvajanje v peskovniku kar nam da možnost varne rabe nepreverjenih predlog.

V našem primeru je predloga dokument HTML z dodanimi kontrolnimi stavki, ki predložnemu stroju povedo kam in kako naj vstavi podatke, na primer seznam novic. Izhodni rezultat je spet datoteka HTML tokrat z vstavljenimi ustreznimi podatki. Sami smo to uporabljali za generiranje vsebine naših gradnikov iz prej pridobljenih podatkov.

Primer predloge:

```
{% extends "base.html" %}
{% block content %}
<ul>
   {% for link, title, date, summary in newsList %}
   <li><a target="_parent" href="{{ link }}">{{ title }}</a></li>
   {% endfor %}
</ul>
{% endblock %}
```

Primer uporabe:

```
def getFriNovice( stNovic=10 ):
   """Novice iz spletne strani fri-ja"""
   template = ENV.get_template('friNovice.html')
   try:
      stNovic = int(stNovic)
   except:
      stNovic = 10
   return template.render(newsList = _getRssNovice()[:stNovic])
```

3.2.5 BeautifulSoup

BeautifulSoup [3] je pythonova programska knjižica, namenjena enostavnemu pridobivanju podatkov iz dokumentov HTML in XML, pogosto so to spletne strani. Njene glavne prednosti so :

- razčleni tudi zelo slabo formiran HTML in pri tem poskuša sestaviti vsaj približno smiselno drevo dokumenta, kar nam skoraj vedno zadostuje, da poberemo želene podatke;
- zagotovi nam enostavne metode za iskanje in navigacijo po drevesu dokumenta;
- za vsak dokument poskuša ugotovi njegovo znakovno kodiranje in ga avtomatsko pretvori v Unicode.

Knjižico BeautifulSoup smo uporabili za pridobivanje podatkov o urnikih in laboratorijih s spletne strani fakultete.

Primer uporabe:

```
def _parseUrnik( urnikUrl ):
    """Sparsamo urnik na podanem linku"""
    urnik = [] # urnik kot 2d tabela

    page = urllib2.urlopen(urnikUrl)
    soup = BeautifulSoup(page)
    trs = soup('tr')[1:] #dobimo vrstice oz. ure na urniku

    for tr in trs[:14]: #gremo po vrsticah, urah dneva
        tds = tr.findAll('td')[:6] #za vsako vrstico dobimo stolpce oz dneve pon..pet
        predmeti = []
        for td in tds:
            ime = td.string if td.string else td.contents[0].string
            dodatno = ",".join(a.text for a in td.findAll('font')[1:])
            predmeti.append([ime.strip(), dodatno])
        urnik.append(predmeti)

    return urnik
```

3.2.6 Universal feed parser

Universal feed parser je knjižica za Python, ki nam omogoča enostavno branje spletnih virov RSS ali Atom. Modul je ena sama datoteka z poglavitno funkcijo „parse“. Ta funkcija prebere spletni vir, do katerega pot ji podamo kot prvi argument, vrne pa podatkovno strukturo, ki je razčlenjen spletni vir. Sedaj lahko do podatkov dostopamo tako kot v slovarju z uporabo ustreznih ključev ali preko klicev funkcij [21].

To knjižico smo uporabili za branje spletnega vira novic iz strani fakultete, kot vidimo na spodnjem primeru:

```
def _rssNovice( url ):
    result = []
    rssFeed = feedparser.parse( url )

    for entry in rssFeed.entries[:20]:
        date = datetime.strptime(entry.updated[:16], "%a, %d %b %Y").strftime("%d. %b %Y")
        date[1:] if date[0]=='0' else date
        result.append( (entry.link, entry.title, date, entry.summary[3:-4]) )
  return result
```

4 Izvedba gradnikov

4.1 Strežniški del

Za samo pripravo podatkov in vsebino gradnikov na strežniški strani skrbita dva pythonova modula, »fricron« in »frisplet«, ki smo ju razvil prav v ta namen.

4.1.1 Modul »fricron«

Modul imenovan »fricron« skrbi za periodično osveževanje podatkov, pridobljenih s spletne strani fakultete in jih shranjuje v lokalno podatkovno bazo. Za njegovo periodično poganjanje skrbi program Cron, ki je prisoten v večini na Unixu temelječih sistemov, kamor spada tudi Linux. Da bo Cron poganjal ta program, je bilo potrebno urediti datoteko crontab, kar storimo z ukazom

crontab –e

V datoteko je potrebno dodati zapis, ki pove kateri ukaz naj se izvede in kako pogosto. V našem primeru je zapis sledeč:

0 * * * * python /var/www/test/fricron.py

Prvi del nam pove, da se bo izvajanje prožilo vsako polno uro, drugi del pa je klic python interpreterja in pot do naše skripte.

4.1.2 Modul »frisplet«

Modul »frisplet« skrbi za generiranje vsebine HTML naših gradnikov.

Ob prejemu zahtevka spletni strežnik Apache, oziroma njegova razširitev mod_python preko »publisher handlerja« poskrbi za nalaganje našega modula in klic ustrezne funkcije, ki nadalje poskrbi za obdelavo zahtevka.

Sama obdelava zahtevka poteka v treh korakih.

V prvem koraku funkcija, ki jo je poklical mod_python, preveri prejete argumente, in če so veljavni, nadaljuje nalaganje ustrezne predloge ter zatem pokliče še ustrezno funkcijo za pridobitev podatkov. V kolikor pa so prejeti parametri neveljavni, uporabniku vrne opozorilo o napaki.
V drugem koraku poklicana funkcija izvede poizvedbo SQL, ali več njih, ter primerno obdelane in strukturirane podatke vrne nazaj kličoči funkciji.
V zadnjem, tretjem koraku se podatki posredujejo predložnemu stroju, ki na podlagi prej naložene predloge ustvari dokument HTML, ki se vrne odjemalcu. Sama predloga je združek kode HTML in kontrolnih stavkov, namenjenih predložnemu stroju.

4.2 Viri podatkov

Podatke, ki predstavljajo vsebino naših gradnikov črpamo iz treh virov.

4.2.1 Spletna stran fakultete

Na spletni strani fakultete dobimo podatke o urnikih, novice in podatke o laboratorijih ter njihovih članih.

Novice so na voljo v obliki spletnega vira RSS ki ga enostavno beremo z uporabo pythonove knjižice „feedparser“.
Pri branju urnikov je nekaj več težav. Najprej je potrebno poiskati ter nato še prebrati in razčleniti ustrezno spletno stran, ki predstavlja urnik. Branje urnikov zato poteka v dveh korakih. V prvem preberemo in razčlenimo stran na naslovu http://urnik.fri.uni-lj.si/. Tako dobimo povezave na podstrani, kjer so dejanski urniki in podatke, ki povedo, komu je urnik na določeni podstrani namenjen. V drugem koraku lahko sedaj preberemo in razčlenimo ustrezno podstran z zahtevanim urnikom.
Podobno poteka tudi branje podatkov o laboratorijih in njihovih članih. Najprej je potrebno prebrati in razčleniti uvodno stran z navedenimi vsemi laboratoriji in povezavami do njihovih podstrani. Nakar sledi še branje in razčlenjevanje strani posameznih laboratorije, kjer pridobimo podatke o njihovih članih.

Ker so opravila pri branju novic in podatkov o laboratorijih ter njihovih članih dokaj zamudna, jih ne izvajamo vsakič sproti ob prejetju zahtevka, temveč povsem neodvisno od tega. Za to poskrbi prej omenjeni modul „fricron“, ki se izvaja periodično vsako uro.

4.2.2 Anonimizirana podatkovna baza spletne učilnice

V podatkovni bazi spletne učilnice najdemo podatke o predmetih ter z njimi povezanih novicah in dogodkih. Sama spletna učilnica temelji na odprtokodnem sistemu Moodle in uporablja podatkovno bazo MySQL. Med pisanjem diplome nam je bil omogočen dostop do anonimizirane kopije podatkov iz te podatkovne baze.

Za uspešno pridobivanje želenih podatkov je bilo potrebno poznati shemo same podatkovne baze ter pomen posameznih tabel in polj v njih. Tu se je pojavila težava, saj je struktura podatkovne baze zelo slabo dokumentirana. Tako smo porabili kar nekaj časa za zbiranje posameznih delov dokumentacije, ki so nam omogočili razumevanje večjega dela podatkovne baze. Za popolno razumevanje nekaterih delov pa je bilo potrebno preizkušanje in opazovanje kaj se ob določenem dejanju zapiše v podatkovno bazo, ter kam. V ta namen je bilo potrebno namestiti aplikacijo Moodle. Uporabili smo namestitveni paket za okolje Windows, ker le-ta vsebuje vse potrebno za delovanje Moodla in njegovo enostavno poganjanje.

Za pregledovanje baze tekom raziskovanja sheme le-te smo uporabljali orodje Toad for MySQL, v katerem smo kasneje tudi pisali in testirali poizvedbe, vključene v modul »frisplet«.

4.2.3 Sistem e-študent

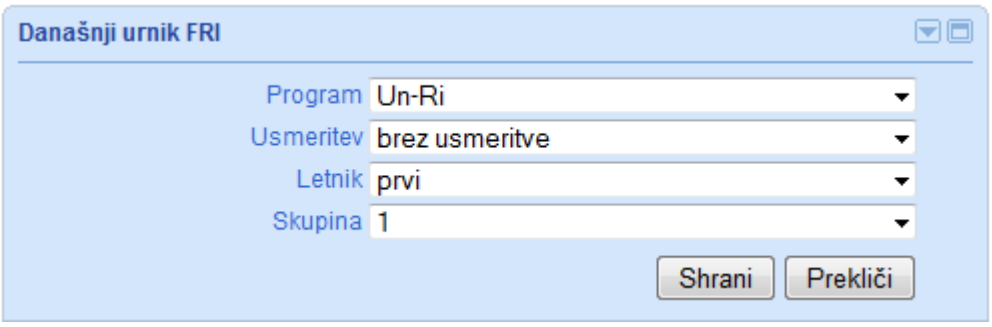
Tretji vir podatkov naj bi bil spletni sistem e-študent, vendar do njega tekom razvoja gradnikov nismo imeli dostopa. Namesto tega smo v lokalni podatkovni bazi ustvarili tabelo z izmišljenimi podatki, ki pa strukturno in vsebinsko ustrezajo tistemu, kar bi nas iz e-Študenta utegnilo zanimati. Kasneje naj bi se nad podatkovno bazo sistema e-študent ustvaril pogled (ang. view), ki bo ustrezal naši predpostavljeni strukturi podatkov.
Za napolnitev začasne tabele, smo napisali skripto, katera to tabelo napolni z naključnimi podatki.

4.3 Izdelani gradniki

Kot rezultat dela je nastalo devet spodaj opisanih gradnikov, katere trenutno najdemo na naslovu http://193.2.72.51/pripomocki.html. Na tej spletni strani je seznam gradnikov, ob vsakem pa je gumb, s klikom katerega izbrani gradnik dodamo na spletno mesto iGoogle. Ko je gradnik dodan, je, da bo prikazoval za nas pravilne podatke, pri nekaterih potrebno urediti še njegove nastavitve.

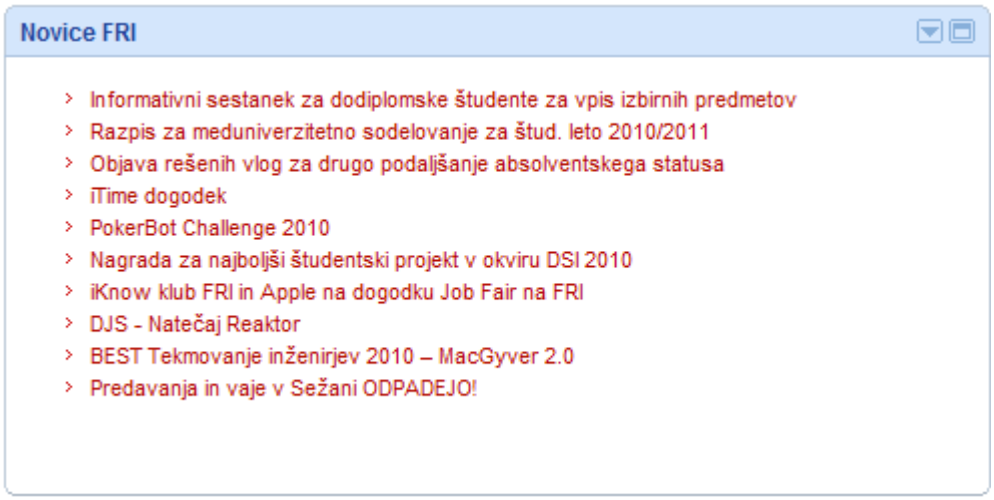
Spodnji sliki 9 in 10 prikazujeta primer nastavitve gradnika, ki prikazuje dnevni urnik.

9. Dostop do nastavitev



10. Primer nastavitev za prikaz urnika

4.3.1 Novice iz spletne strani Fakultete za računalništvo in informatiko

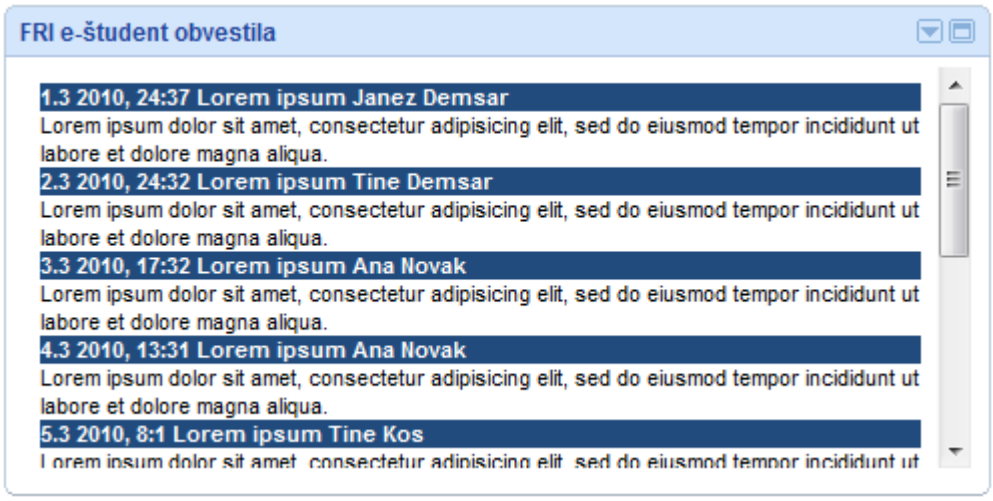


11. Gradnik z seznamom povezav na novice

Gradnik na sliki 11 prikazuje seznam naslovov zadnjih novic, ki so objavljene na spletnem mestu Fakultete za računalništvo in informatiko, pri čemer nas klik na naslov novice odpelje do izvorne novice. Uporabnik lahko nastavi želeno število prikazovanih novic.

Priprava podatkov za ta gradnik poteka v dveh korakih. Prvi korak je branje spletnega vira RRS in njegovo razčlenjevanje, rezultat le tega pa zapišemo v podatkovno bazo. Ta korak je naloga modula „fricron“ in se izvaja periodično vsako uro, za kar skrbi program „Cron“. Drugi korak se izvede ob prejetju zahtevka in takrat iz podatkovne baze preberemo prej pripravljene podatke ter tvorimo HTML vsebino našega gradnika.
Razdelitev v dva koraka je potrebna zaradi zmanjševanja odzivnega časa ter obremenitve in porabe pasovne širine spletnega strežnika iz katerega pobiramo novice.

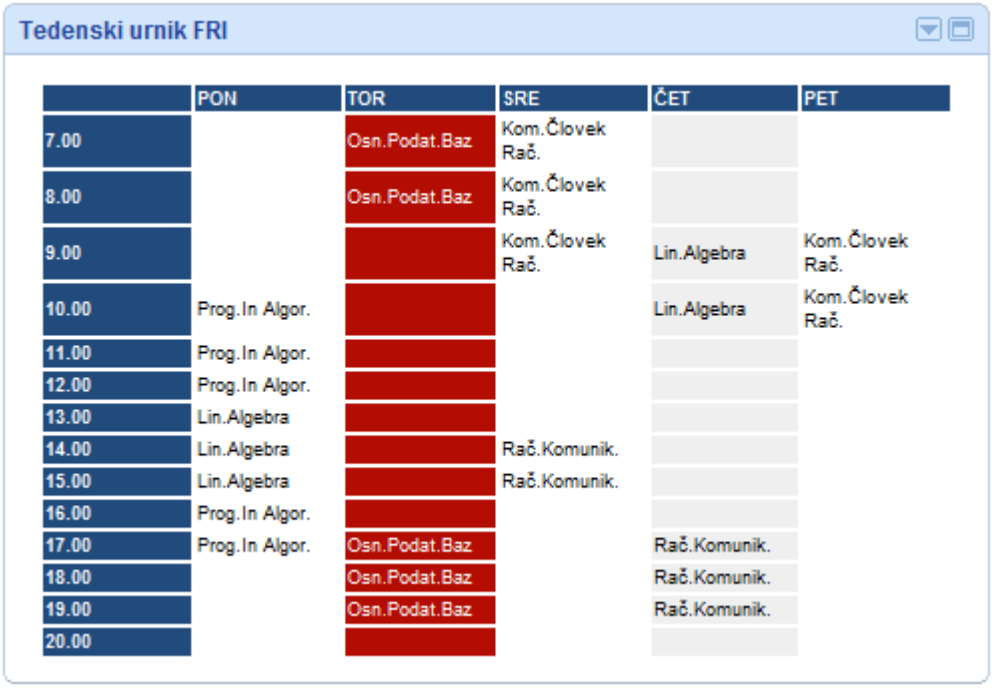
4.3.2 Obvestila iz sistema e-študent



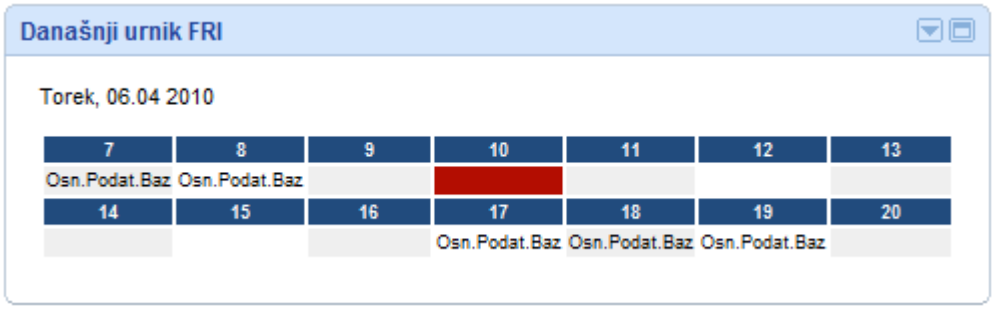
12. Gradnik z obvestili iz sistema e-študent

Gradnik na sliki 12 je namenjen prikazu obvestil, ki jih študentje prejemamo v sistem e-študent, pri čemer je moč nastaviti datum zadnje novice.

4.3.3 Tedenski urnik in dnevni urnik



13. Gradnik z tedenskim urnikom



14. Gradnik z dnevnim urnikom

Gornja gradnika prikazujeta podatke o urniku pri čemer gradnik na sliki 13 prikazuje tedenski, na sliki 14 pa dnevni urnik za določenega študenta. Za prikaz pravilnega urnika mora le ta v nastavitvah gradnikov izbrati ustrezen program, usmeritev, letnik in skupino. Za boljšo preglednost se pri tedenskem urniku rdeče obarva stolpec, ki ustreza trenutnemu dnevu, pri dnevnem pa stolpec, ki ponazarja trenutno uro.

Podatke o urnikih, dobimo z razčlenjevanjem spletne strani http://urnik.fri.uni-lj.si/ na kateri se nahajajo urniki za študente FRI. Od prejetju zahtevka se izvedejo trije koraki. V prvem koraku pridobimo seznam povezav na vse urnike, nato v drugem koraku na podlagi nastavitev gradnika izberemo ustrezen urnik in z razčlenitvijo pridobimo podatke. Ta dva koraka sta enaka za obagradnika, razlika je v tretjem koraku kjer se tvori njuna html vsebina.

4.3.4 Gradniki, ki se navezujejo na spletno učilnico

Tukaj so opisani štirje gradniki, kateri se navezujejo na spletno učilnico. Pri vseh mora uporabnik v nastavitvah vnesti vpisno številko in geslo, kot ga ima v spletni učilnici. Vsi razen gradnika, ki prikazuje seznam predmetov na katere smo prijavljeni, omogočajo tudi nastavitev števila prikazanih vnosov.

Podatke za vse gradnike trenutno pridobimo iz anonimizirane kopije podatkov baze spletne učilnice, kasneje pa se bomo povezali na pravo podatkovno bazo.

Novice s spletne učilnice Fakultete za računalništvo in informatiko

15. Gradnik z novicami iz spletne učilnice

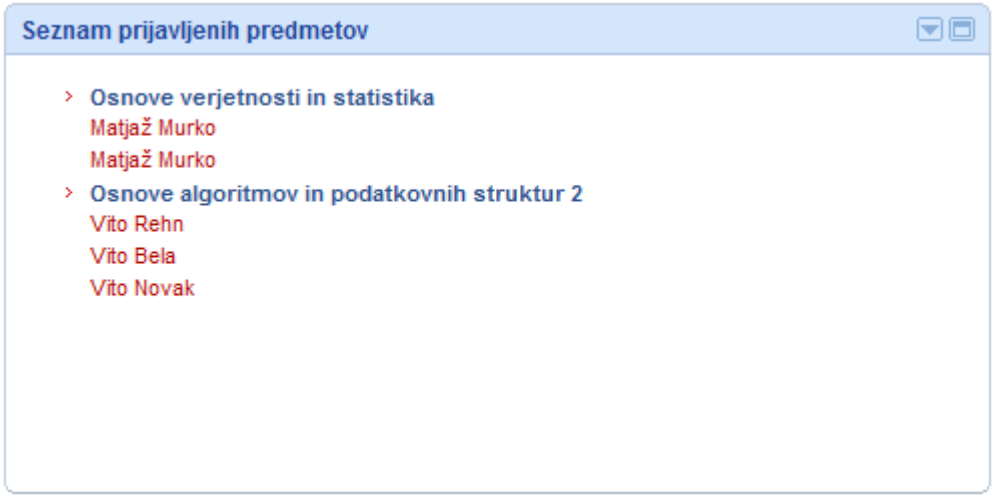
Gradnik na sliki 15 prikazuje povezave na novice, datum objave le-te in ime predmeta kjer je bila objavljena in sicer za predmete na katere je študent prijavljen v spletni učilnici.

Odprti kvizi s spletne učilnice

16. Gradnik s seznamom odprtih kvizov na spletni učilnici

Gradnik na sliki 16 prikazuje povezave na kvize, datum objave le-te in ime predmeta pri katerem so bili odprti in sicer za predmete na katere je študent prijavljen v spletni učilnici.

Seznam predmetov na katere smo prijavljeni ter njihovi predavatelji in asistenti



17. Gradnik s seznamom predmetov na katere smo prijavljeni v spletni učilnici

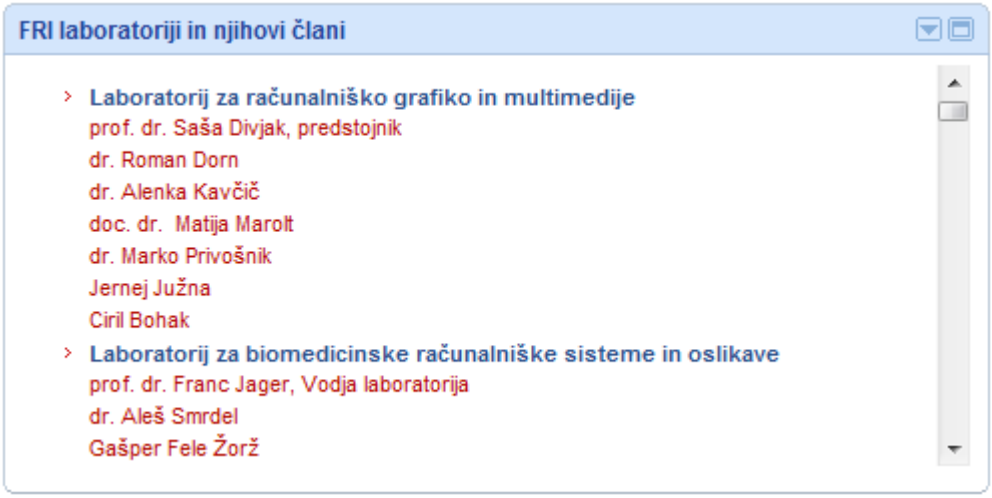
Gradnik na sliki 17 prikazuje seznam predmetov na katere smo prijavljeni ter za vsak predmet tudi njegove predavatelje in asistente. Klik na ime predmeta nas popelje na njegovo stran na spletni učilnici, klik na ime predavatelja pa na stran le-tega.

Zadnji dogodki v prijavljenih predmetih

18. Gradnik s seznamom zadnjih novic na spletni učilnici

Gradnik na sliki 18 prikazuje povezave na zadnje dogodke, datum objave le-teh in ime predmeta h kateremu dogodek spada in sicer za predmete na katere je študent prijavljen v spletni učilnici.

4.3.5 Seznam laboratorijev in njihovih članov



19. Gradnik s seznamom laboratorijev in njihovih članov

Gradnik na sliki 19 prikazuje seznam vseh laboratorijev in njihovih članov. Klik na ime laboratorija ali osebe nas odpelje na njegovo domačo stran.

Podatke za gradnik pripravimo v dveh korakih. Prvi korak je naloga modula „fricron“ in vključuje razčlenjevanje spletne strani z seznamom laboratorijev in njihovih članov.
Drugi korak se izvede ob prejetju zahtevka in takrat iz podatkovne baze preberemo prej pripravljene podatke ter tvorimo HTML vsebino našega gradnika.
Razdelitev v dva koraka je potrebna zaradi zmanjševanja odzivnega časa ter obremenitve in porabe pasovne širine spletnega strežnika iz katerega pobiramo podatke.

5 Zaključek

Kot rezultat pričujočega diplomskega dela je tako nastalo devet gradnikov, ki bodo služili študentom naše fakultete kot pripomoček za lažji dostop do različnih informacij. Ti so:

- novice FRI,
- FRI e-študent obvestila,
- Tedenski urnik FRI,
- Današnji urnik FRI,
- Novice iz spletne učilnice FRI,
- Odprti kvizi na spletni učilnici,
- Seznam prijavljenih predmetov,
- Zadnji dogodki na spletni učilnici,
- FRI laboratoriji in njihovi člani.

Sami gradniki so tipa URL, kar pomeni da se vsebina in programska logika ne nahaja v sami specifikaciji gradnika, ampak le-ta vsebuje zgolj povezavo do skripte, ki vsebino generira. Za izvajanje te skripte ter pretvarjanje URL naslova v klic ustrezne funkcije, katera generira vsebino gradnika, pa poskrbi Mod_Python kot razširitveni modul strežnika Apache . Informacije, ki nam jih gradniki nudijo pridobimo iz treh različnih virov in sicer s spletne strani fakultete, sistema e-študent in spletne učilnice.

V nadaljevanju se bo s strani študentov verjetno pokazala potreba po razvitju še dodatnih gradnikov, možno pa je tudi kar nekaj izboljšav že obstoječih. Glavna izboljšava bi tako bila prehod na HTML tip gradnikov, s čimer pridobimo možnost uporabe JavaScript funkcij, ki jih prinaša programski umesnik gradnikov, ter s tem boljšo uporabniško izkušnjo. Sicer naj bi te funkcije lahko uporabljali tudi pri URL tipu, a v praksi naletimo na mnogo težav za katere ni neke prave rešitve, razen da se svetuje uporaba HTML tipa.

6 Viri

[1] (2010) What are gadgets. http://code.google.com/apis/gadgets/docs/overview.html

[2] (2010) Gadgets API. Dostopno na: http://code.google.com/intl/sl/apis/gadgets/

[3] (2010) Beautiful Soup. Dostopno na: http://www.crummy.com/software/BeautifulSoup/

[4] (2010) Jinja2. Dostopno na: http://jinja.pocoo.org/2/documentation/

[5] (2010) VMware Workstation 7. Dostopno na: http://www.vmware.com/products/workstation/

[6] (2010) VMware. Dostopno na: http://en.wikipedia.org/wiki/VMware

[7] (2010) Ubuntu Forums. Dostopno na: http://ubuntuforums.org/

[8] (2010) Ubuntu Home Page. Dostopno na: http://www.ubuntu.com/

[9] (2010) Apache HTTP Server.Dostopno na: http://en.wikipedia.org/wiki/Apache_HTTP_Server

[10] (2010) The Apache http Server Project. Dostopno na: http://httpd.apache.org/

[11] (2010) Mod_python Manual Dostopno na: http://www.modpython.org/live/current/modpython.pdf

[12] (2010) MySQL Dostopno na: http://en.wikipedia.org/wiki/MySQL

[13] (2010) Writing MySQL Scripts with Python DB-API. Dostopno na: http://www.kitebird.com/articles/pydbapi.html

[14] (2010) MySQLdb User's Guide. Dostopno na: http://mysql-python.sourceforge.net/MySQLdb.html

[15] (2010) SCP. Dostopno na: http://en.wikipedia.org/wiki/Secure_copy

[16] (2010) WinSCP. Dostopno na: http://en.wikipedia.org/wiki/WinSCP

[17] (2010) Python za programerje. Dostopno na: http://ucilnica.fri.uni-lj.si/mod/resource/view.php?id=3834

[18] (2010) Python. Dostopno na: http://en.wikipedia.org/wiki/Python_%28programming_language%29

[19] (2010) Eclipse. Dostopno na: http://en.wikipedia.org/wiki/Eclipse_%28software%29

[20] (2010) Toad for MySQL. Dostopno na: http://en.wikipedia.org/wiki/TOAD_%28software%29

[21] (2010) Universal Feed Parser. Dostopno na: http://www.feedparser.org/docs/

[22] (2010) SSH. Dostopno na: http://en.wikipedia.org/wiki/Secure_Shell

[23] (2010) PuTTy. Dostopno na: http://en.wikipedia.org/wiki/PuTTY

[24] (2010) Moodle. Dostopno na: http://moodle.org/